INTERFAZ GRAFICA DE USUARIO PARA LA INTERACCION CON LOS MODULEOS DEL SCADA SAINUX (VSERSAI)

Ing. Luis Andrés Valido Fajardo[1]

1. Universidad de Matanzas – Sede "Camilo Cienfuegos"- Departamento de Desarrollo de Recursos para Aprendizaje, Vía Blanca Km.3, Matanzas, Cuba. luis.valido@umcc.cu

Resumen

SAINUX (Sistema de Automatización Industrial basado en GNU/LINUX), es un sistema SCADA distribuido, orientado a componentes y en proceso de desarrollo por el Centro de Informática Industrial (CEDIN) de la Universidad de las Ciencias Informáticas (UCI). Para los procesos de desarrollo y prueba del módulo HMI perteneciente al SAINUX se hace indispensable que el desarrollador o probador según sean el caso tenga en su estación de trabajo el resto de los módulos que componen el SCADA SAINUX ejecutándose como procesos del sistema operativo para poder desarrollar su labor. El presente trabajo tiene como objetivo principal el diseño y desarrollo de una interfaz de usuario para la interacción con los módulos del SCADA SAINUX. En el trabajo se utilizaron métodos científicos como el analítico-sintético, histórico-lógico y observación. Además, se utilizó *Extreme Programming* como metodología de desarrollo, lenguaje de modelado UML, la herramienta CASE *Visual Paradigm*, el lenguaje de programación *Python* y como marco de desarrollo Qt. Como resultado se obtuvo un sistema que proporciona una interfaz de usuario basada en interfaces gráfica para interactuar con los módulos del SCADA SAINUX.

***Palabras claves**: Interfaz gráfica, Interfaz de usuario, SCADA*

ISBN: 978-959-16-4235-6

Introducción

Un sistema de supervisión, control y adquisición de datos SCADA se refiere a un sistema central que monitorea[1] y controla un sitio completo o un sistema que se extiende sobre una gran distancia (kilómetros / millas). La instalación de un sistema SCADA necesita un hardware[2] de señal de entrada y salida, sensores[3] y actuadores, controladores, HMI, redes, comunicaciones, base de datos entre otros (Penin, 2012).

Los sistemas SCADA se componen por los siguientes módulos:

1. **Manejadores:** Aseguran mediante una interfaz genérica la comunicación del sistema de supervisión y control con los distintos dispositivos que existen en el campo, ya sean autómatas, sensores inteligentes, etc.
2. **Núcleo de Procesamiento de Datos:** Representa el núcleo principal del procesamiento de los datos, es el encargado del procesamiento y análisis de la información recogida del campo a través de los manejadores. Una vez procesada esta información, es enviada al módulo que la requiera.
3. **Base de Datos Históricos:** Aquí se implementa el mecanismo encargado del almacenamiento de la información recibida desde el campo, así como la sucesión de alarmas y eventos generados. La información almacenada es utilizada por varias aplicaciones del sistema.
4. ***Middleware:*** El *Middleware* es la capa de software, que se encarga de la comunicación entre los diferentes procesos distribuidos de medio y alto nivel, que forman parte del sistema SCADA y el principal elemento que caracteriza su complejidad es la gestión de las comunicaciones.
5. **Interfaz Hombre - Máquina o HMI (*Human Machine Interface*):** El módulo de HMI en el SCADA se encarga de representar, en un ordenador, los procesos que ocurren en el campo, muestra los componentes implicados, los sensores, las estaciones remotas, y el sistema de comunicación dándole al operador total control. Éste módulo es el que permite al operador estar en contacto directo con el sistema, realizar la supervisión y el control del proceso en general. Está compuesto por dos partes fundamentales:
    a. **Entorno de configuración (EC) o Editor:** Permite configurar varios procesos o partes de ellos, aquí se definen y gestionan las variables, los manejadores, los comandos, las alarmas y variadas opciones adicionales. Este entorno funciona como una aplicación de diseño tradicional, con la peculiaridad que los sinópticos[4] se confeccionan a partir de objetos y

[1] Observa el curso de uno o varios parámetros para detectar posibles anomalías.
[2] Corresponde a todas las partes físicas y tangibles de una computadora, sus componentes eléctricos, electromecánicos y mecánicos; sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado.
[3] Dispositivo capaz de transformar magnitudes físicas o químicas en magnitudes eléctricas
[4] Que presenta las partes principales de un asunto de manera clara, rápida y resumida.

ISBN: 978-959-16-4235-6

primitivas básicas predefinidas, que se pueden agrupar, combinar, transformar, importar y exportar entre otras.

b. **Entorno de visualización (EV) o Visualizador:** Se encarga de visualizar las animaciones y los objetos definidos en el editor, muestra lo que está ocurriendo en el campo en tiempo real, es el que envía los comandos a las estaciones remotas, quién recibe los valores de las variables, interactúa con la mayoría de los operadores pues se emplea para supervisar el proceso de manera directa.

La informatización de la sociedad cubana es una voluntad política del gobierno que está expresada en los Lineamientos de la Política Económica y Social, aprobados en el Sexto Congreso del Partido Comunista de Cuba (2011). En varios sectores de la industria y la economía cubana se necesita un sistema capaz de informatizar la supervisión y control de sus procesos. SAINUX (Sistema de Automatización Industrial basado en GNU/LINUX), es un sistema SCADA distribuido, orientado a componentes y en proceso de desarrollo por especialistas del Centro de Informática Industrial (CEDIN) de la Universidad de las Ciencias Informáticas (UCI).

En la actualidad para los procesos de desarrollo y prueba del módulo HMI se hace indispensable que el desarrollador[5] o probador[6] según sean el caso tenga en su estación de trabajo el resto de los módulos que componen el SCADA SAINUX ejecutándose como procesos del sistema operativo para poder desarrollar su labor debido a que un gran porciento de las funcionalidades que brinda o se desarrollan en dicho módulo dependen de la interacción de él con el restos de los módulos.

Para iniciar, reiniciar o simplemente detener el resto de los módulos se hace a través de una consola con una secuencia de comandos o se crean un grupo de ficheros en lenguaje BASH (Garrels, 2010) con la secuencia de comandos para realizar una de las acciones anteriormente mencionadas los cuales son ejecutados de similar forma desde una consola. Lo expresado trae como consecuencia que el trabajo desplegado por algún integrante de los equipos de desarrollo o prueba del HMI es más complejo y engorroso de lo esperado producto a la interacción que debe establecer con una interfaz de consola las cual tiene como desventajas:

- No son fáciles de operar.
- Hay que tener presente la funcionalidad de los comandos.

[5] Este rol es responsable de desarrollar las funcionalidades del sistema, incluyendo su diseño para ajustarlo en la arquitectura, implementando, probando, integrando y documento componentes que hagan parte de la solución.

[6] Este rol es responsable de las actividades principales del esfuerzo de las pruebas. Estas actividades incluyen identificar, definir, implementar y dirigir las pruebas necesarias, como también verificar los resultados de las pruebas y analizar los resultados.

ISBN: 978-959-16-4235-6

- No son atractivas hacia el usuario.

- Requieren por parte del usuario una mayor experiencia.

Por todo lo anteriormente descrito se define como problema de investigación ¿Cómo contribuir a la interfaz de usuario para la interacción entre los desarrolladores y probadores del módulo HMI con el resto de los módulos del SCADA SAINUX? Por lo que se plantea como objetivo general el diseño y desarrollo de una interfaz de usuario para la interacción con los módulos del SCADA SAINUX.

Materiales y métodos o Metodología computacional

Conceptos asociados a la investigación

Interfaz de usuario: La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar. Dentro de las interfaces de usuario se puede distinguir básicamente tres tipos (Royo, 2004):

- **Una interfaz de hardware:** A nivel de los dispositivos utilizados para ingresar, procesar y entregar los datos: teclado, ratón y pantalla visualizadora.

- **Una interfaz de software:** Destinada a entregar información acerca de los procesos y herramientas de control, a través de lo que el usuario observa habitualmente en la pantalla.

- **Una interfaz de software-hardware:** Establece un puente entre la máquina y las personas, permite a la máquina entender la instrucción y a el hombre entender el código binario traducido a información legible.

Atendiendo a como el usuario puede interactuar con una interfaz, nos encontramos con varios tipos de interfaces de usuario (Shneiderman, 2010):

- Interfaces alfanuméricas (intérpretes de comandos) que solo presentan texto.

- Interfaces gráficas de usuario (GUI, *graphic user interfaces*), las que permiten comunicarse con el ordenador de una forma muy rápida e intuitiva representando gráficamente los elementos de control y medida.

- Interfaces táctiles, que representan gráficamente un panel de control en una pantalla sensible que permite interactuar con el dedo de forma similar a si se accionara un control físico.

ISBN: 978-959-16-4235-6

SCADA SAINUX: El sistema SCADA SAINUX es un sistema distribuido, compuesto por diferentes subsistemas o módulos que trabajan de manera conjunta para llevar a cabo las tareas de supervisión, control y adquisición de datos. Cada uno de estos módulos es encargado de realizar una tarea específica:

Comunicaciones: Se encarga de la comunicación entre los diferentes módulos que forman parte del sistema. Implementa dos formas de comunicación, el modelo de comunicación asincrónica (Moore, 1992), el cual tiene como base el Servicio de Notificación de TAO, fundamentado en el paradigma publicación-subscripción y el modelo sincrónico (Hines, 2008) inherente al modelo cliente/servidor de la especificación utilizada (CORBA) (Gelbmann, 2002).

Configuración: El módulo de configuración está formado por varios componentes o elementos de la configuración del proyecto relacionado con seguridad, comunicación, con los dispositivos de campo, variables entre otros. Es el encargado de almacenar, persistir y suministrar la información base para el funcionamiento de los demás módulos del SCADA.

Adquisición o núcleo de procesamiento de datos: El núcleo de procesamiento de datos es el encargado del tratamiento y análisis en tiempo real de la información adquirida desde el campo.

Seguridad: Este módulo proporciona las funcionalidades necesarias para garantizar el trabajo autorizado a usuarios y módulos. Tiene implementada herramientas que protegen al sistema de ataques maliciosos o involuntarios por parte de personas o recursos, tales como fallas de energía y problemas de red.

Histórico o BDH: Se encarga del almacenamiento y procesado ordenado de los datos, de forma que otra aplicación o dispositivo pueda tener acceso a ellos, para realizar posteriores análisis como diagnósticos o reportes.

Interfaz Hombre Máquina: Este módulo se encarga de representar en un ordenador, los procesos que ocurren en el campo en tiempo real. Muestra de forma gráfica los procesos operacionales, alarmas, variables, reportes y despliegues que permiten al operador el contacto directo con el sistema.

Soluciones de software analizadas

Panel de control de XAMPP: El panel de control de XAMPP le da un control completo sobre todos los componentes de XAMPP instalados. Puede usar el panel de control para iniciar/parar distintos módulos, lanzar una ventana de comandos UNIX, abrir el explorador de Windows, y ver todas las operaciones que se ejecutan en segundo plano (Labrador, 2008).

ISBN: 978-959-16-4235-6

Administrador de tareas: Es un programa informático que se utiliza para proporcionar información sobre los procesos y programas que se están ejecutando en una computadora y su situación general. Puede emplearse para finalizar procesos, comprobar el uso de CPU, así como terminar programas y cambiar la prioridad entre procesos (Mueller, 2002).

Webmin: Es una herramienta de configuración de sistemas accesible vía web para OpenSolaris, GNU/Linux y otros sistemas Unix. Con él se pueden configurar aspectos internos de muchos sistemas operativos, como usuarios, cuotas de espacio, servicios, archivos de configuración, apagado del equipo, etcétera, así como modificar y controlar muchas aplicaciones libres, como el servidor web Apache, PHP, MySQL, DNS, Samba, DHCP, entre otros (Karzynski, 2014).

Kaspersky Anti-Virus: Es un antivirus que realiza una excelente combinación de protección reactiva y preventiva, protegiéndote eficazmente de virus, troyanos y todo tipo de programas malignos. Adicionalmente, dentro del grupo de programas malignos, Kaspersky Anti-Virus también se encarga de proteger el sistema contra programas potencialmente peligrosos como los spyware. Cuenta con una interfaz de usuario basada en ventanas desde las cuales los usuarios pueden iniciar o detener los servicios que brinda esta herramienta (2015).

Métodos teóricos

Analítico-Sintético: Se utiliza para analizar teorías y elementos bibliográficos relacionados con las interfaces de usuarios, permitiendo la extracción de los elementos más importantes que hacen viable el comienzo de la investigación.

Histórico-Lógico: Se utiliza para el estudio del fenómeno en cuestión y su evolución histórica, permitiendo así, profundizar los conocimientos sobre el tema.

Métodos empíricos

Observación: Se emplea para estudiar las características y comportamientos de las soluciones similares, permitiendo obtener información relevante sobre el funcionamiento de dichas soluciones.

Metodología de desarrollo de software: *Extreme Programming*

Con una buena metodología se pretende reducir costos y retrasos de proyectos, así como mejorar la calidad del software. *Extreme Programming* o Programación Extrema es una de las llamadas metodologías ágiles de desarrollo de software más exitosas y controversiales de los tiempos recientes. *Extreme Programming* (XP) surge como una nueva manera de encarar proyectos de software, proponiendo una metodología basada esencialmente en la simplicidad y agilidad (Beck, 2002). La metodología propuesta en XP está diseñada para

ISBN: 978-959-16-4235-6

entregar el software que los clientes necesitan en el momento en que lo necesitan. Si bien el ciclo de vida de un proyecto XP es muy dinámico, se puede separar en fases.

- **Fase de exploración**. Es la fase en la que se define el alcance general del proyecto. En esta fase, el cliente define lo que necesita mediante la redacción de sencillas historias de usuarios. Los programadores estiman los tiempos de desarrollo en base a esta información.

- **Fase de planificación**. La planificación es una fase corta, en la que el cliente, los gerentes y el grupo de desarrolladores acuerdan el orden en que deberán implementarse las historias de usuario, y, asociadas a éstas, las entregas.

- **Fase de iteraciones**. Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración.

- **Fase de puesta en producción**. Si bien al final de cada iteración se entregan módulos funcionales y sin errores, puede ser deseable por parte del cliente no poner el sistema en producción hasta tanto no se tenga la funcionalidad completa.

Lenguaje de Modelado: UML

UML (*Unified Modeling Language*) es desde finales de 1997 un lenguaje de modelado visual que se utiliza para especificar, visualizar, construir y documentar artefactos de un sistema de software (Rumbaugh, 2007).

Es válido destacar que UML es un lenguaje de modelado, no un método o un proceso. UML constituye un lenguaje más expresivo, claro y uniforme que los anteriores definidos para el diseño Orientado a Objetos, no brinda el total éxito de los proyectos pero si mejora sustancialmente el desarrollo de los mismos al posibilitar una nueva y fuerte integración entre las herramientas, los procesos y los dominios.

Herramienta CASE: *Visual Paradigm*

Herramienta CASE (del inglés, *Computer Aided Software Engineering*) con licencia gratuita, que propicia un conjunto de ayudas para el desarrollo de programas informáticos dando soporte al modelado visual con UML (del inglés, *Unified Modeling Language*), desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Oscar, 2013).

Lenguaje de programación: *Python*

ISBN: 978-959-16-4235-6

Es un lenguaje muy expresivo, es decir, los programas *Python* son muy compactos: un programa *Python* suele ser más corto que su equivalente en lenguajes como C. *Python* llega a ser considerado por muchos un lenguaje de programación de muy alto nivel. La sintaxis de *Python* es muy elegante y permite la escritura de programas cuya lectura resulta más fácil que si se utilizaran otros lenguajes de programación.

Python ofrece un entorno interactivo que facilita la realización de pruebas y ayuda a despejar dudas acerca de ciertas características del lenguaje. Su entorno de ejecución detecta muchos de los errores de programación que escapan al control de los compiladores y proporciona información para detectarlos y corregirlos (Marzal, 2002).

Módulo psutil

Este módulo brinda una interfaz para la recolección de información de todos los procesos en ejecución y la utilización del sistema (procesador, memoria, red) de una manera sencilla utilizando *Python*, con funcionalidades ofrecidas por herramientas como *ps*, *top*, *nice*, entre otras. Actualmente tiene soporte para los sistemas operativos Windows, GNU/Linux, OSX y Free-BSD.

Framework de desarrollo: Qt

El *framework* (marco de trabajo) de desarrollo Qt es un conjunto de herramientas multiplataforma utilizada para construir aplicaciones en C++, *Python* con interfaces gráficas o no en dependencia de las necesidades del desarrollador. Está patentado bajo la licencia GPL 2(Marcos, 2006) y LGPL (Lombardo, 2001) . Actualmente Qt es utilizado en aplicaciones para dispositivos móviles y aplicaciones de escritorio para computadoras.

Más de 500 000 desarrolladores de todo el mundo lo utilizan y entre las empresas que elaboran productos con esta biblioteca se encuentran *Jolla, Navico, ABB, Thales*. Entre las utilidades que brinda esta biblioteca se encuentran las de diseño de interfaces de usuario, dibujar gráficas 2D y 3D de alta calidad utilizadas en diversos ambientes como en la gestión de negocios, herramientas de monitorización entre otras (Blanchette, 2006) .

Requisitos funcionales

Los requisitos funcionales son funciones del sistema de software o sus componentes. Cada función se describe como un conjunto de entradas, comportamientos y salidas. Los requisitos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone, un sistema debe cumplir. Son complementados por los requisitos no funcionales, que se enfocan en cambio en el diseño o la implementación. Establecen los comportamientos del sistema (Sommerville, 2015).

Por lo anteriormente explicado se definen como requisitos funcionales:

ISBN: 978-959-16-4235-6

Ejecutar el sistema como superusuario o *root*.
Operar sobre un módulo (iniciar, reiniciar y detener).
Editar el comando de operación.
Visualizar salida de la operación cuando se realiza desde el sistema.
Guardar salida de la operación.
Seleccionar usuario para operar sobre un módulo.
Mostrar el estado (Ejecución / Detenido) del módulo.
Configurar salida de la operación.

Requisitos no funcionales

Los requerimientos no funcionales son requisitos que imponen restricciones en el diseño o la implementación. Son propiedades o cualidades que el producto debe cumplir. Sin embargo estos requisitos no definen el éxito del producto, pero influyen considerablemente en la evaluación del mismo. Teniendo en cuenta las características del sistema se definieron los siguientes requerimientos no funcionales:

Forma de interacción de la interfaz: Basada en ventanas.
Implementación con tecnologías libres.
Ayuda: El sistema debe proporcionar una ayuda que permita al usuario disponer de información útil para la explotación.
El sistema debe contar con la política marcaria de interfaz única de la institución.

Resultados y discusión

A continuación se muestra el diagrama de clases que refleja las entidades que componen la solución de este trabajo y una breve descripción de cada una de ellas.

ISBN: 978-959-16-4235-6

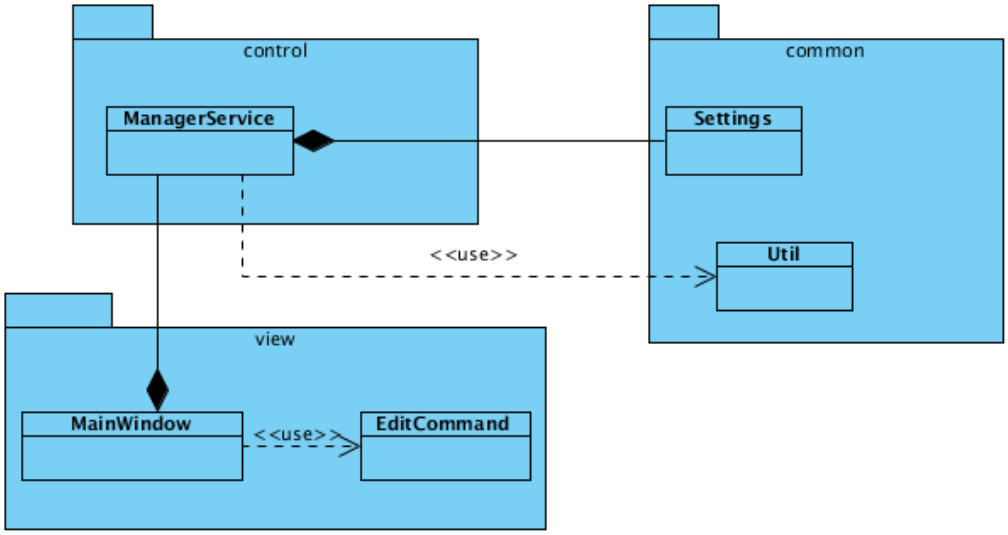

Figura 1. Diagrama de clases de la solución.

Partiendo de la estructura que propone *Python* para las soluciones desarrolladas en dicho lenguaje (Duque, 2011), la solución elaborada está conformada por los siguientes paquetes:

- ***common***: Paquete encargado de agrupar a todos aquellos módulos cuya función es contribuir el correcto funcionamiento del sistema. El mismo lo integran los módulos *Util* y *Settings*, en ambos casos dichos módulos están integrados por una clase cada uno con similar nombre al módulo al que están asociados. La clase *Settings* es la encargada de guardar las configuraciones realizadas por el usuario antes de cerrar el sistema con el objetivo que en la próxima ejecución del programa el mismo inicie con dicha configuración. Mientras el objetivo de la clase *Util* es la de proporcionar funcionalidades de apoyo a las operaciones realizadas por el sistema.
- ***control***: Paquete que aglomera los módulos encargados de realizar la lógica de negocio del sistema. En dicho paquete se encuentra el módulo *ManagerService* al cual pertenece la clase *ManagerService* cuyo objetivo es como bien lo indica su nombre es realizar operaciones sobre los módulos, chequear el estado de los mismos, gestión de los comandos para realizar las operaciones sobre los módulos.
- ***view***: Paquete responsable de congregar los módulos que conforman la interfaz de usuario del sistema. Dentro de los módulos que componen dicho paquete está *MainWindow* compuesto por la clase de similar nombre encargada de la interfaz principal del sistema con la cual el usuario va a tener la mayor interacción. El otro módulo que presente en el paquete es *EditCommand* cuya única clase con igual nombre se encarga de visualizar una interfaz gráfica donde el usuario podrá editar la secuencia de comandos para operar sobre un módulo del SCADA SAINUX.

En las siguientes figuras se muestran las principales interfaces gráficas con cuenta el sistema:

ISBN: 978-959-16-4235-6

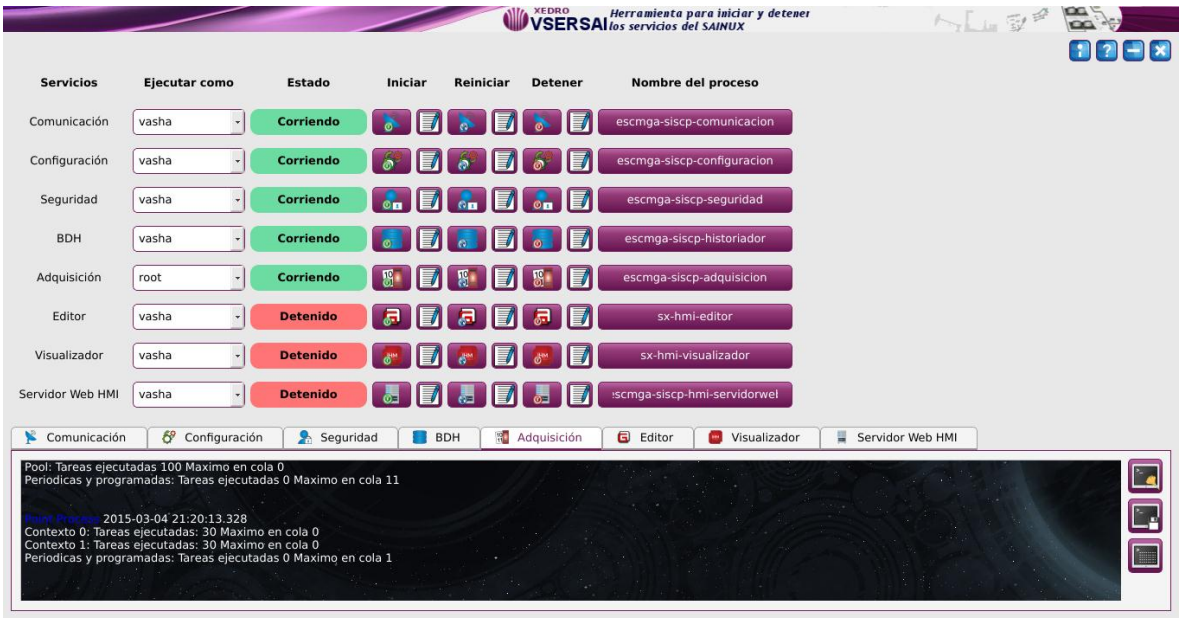

Figura 2. Interfaz gráfica principal del sistema.

Figura 3. Interfaz gráfica para editar el comando para operar sobre un módulo del SCADA SAINUX.

Entre los logros que se obtienen con la solución expuesta en este trabajo se encuentran la reducción de tiempo por parte del usuario para modificar el estado un módulo quedando reflejados en la siguiente tabla.

Módulo	Consola	VSERSAI
Comunicación	4 seg	1 seg
Configuración	3 seg	1 seg

ISBN: 978-959-16-4235-6

Seguridad	3 seg	1 seg
BDH	3 seg	1 seg
Adquisición	5 seg	1 seg
HMI	4 seg	1 seg

Tabla 1. Tiempo que necesita el usuario para modificar el estado de cada uno de los módulos.

Otro de los resultados logrado con la realización de este trabajo es la nueva variante que se le brinda a los equipos de prueba y de HMI para interactuar con el resto de los módulos del SCADA SAINUX a través de una interfaz de usuario basada en interfaces gráficas la cual tiene como elementos positivos a su favor que permite interactuar al usuario con los módulos del SCADA SAINUX de una forma más fácil, proporciona mecanismos estándar de control como botones, ventanas y cuadros de diálogo, cada módulo así como su estado puede ser visualizado en la pantalla mediante una imagen que lo representa, su uso es intuitivo.

Conclusiones

La realización del presente trabajo posibilitó desarrollo del sistema VSERSAI que contribuye a la interfaz de usuario usada por los integrantes de los equipos de prueba y desarrollo del módulo HMI para interactuar con el resto de los módulos del SCADA SAINUX. La solución descrita es una variante diferente a las existentes en la actualidad, siendo novedosa por la utilización de interfaces gráfica para la interacción. Reduce el tiempo que necesita por el usuario para la modificación del estado del módulo. Brinda una configuración inicial para interactuar con los módulos pero permite que el usuario realice configuraciones personalizadas las cuales son guardadas, evitando que el usuario tenga que recordar secuencia de comandos en próximas ejecuciones del sistema. Destacar que en el desarrollo del trabajo fueron utilizadas tecnologías y herramientas libres.

Bibliografía

Beck, K. and Molina, J.G. 2002. Una explicación de la programación extrema: aceptar el cambio. 2002.

Blanchette, J. and Summerfield, M. 2006. C++ GUI Programming with Qt 4. 2006.

Duque, Raúl González. 2011. Python para todos. 2011.

Garrels, M. 2010. Bash Guide for Beginners (Second Edition). s.l. : Fultus Corporation, 2010.

Gelbmann, R. 2002. Evaluation and Comparison of CORBA (Object Request Broker) Implementations. 2002.

ISBN: 978-959-16-4235-6

Hines, J. and University of Pittsburgh. 2008. Highly Synchronous Communication - Characterization, Modeling and. s.l. : University of Pittsburgh, 2008.

Karzynski, M. 2014. Webmin Administrator's Cookbook. s.l. : Packt Publishing, 2014.

2015. Kaspersky. s.l. : Micronet, 2015.

Labrador, Ramón M. Gómez. 2008. Taller de Instalación de Servidores LAMP WAMP. 2008.

2011. Lineamientos de la política económica y social del partido y la revolución. 2011. http://www.cubadebate.cu/wp-content/uploads/2011/05/folleto-lineamientos-vi-cong.pdf

Lombardo, J. 2001. Embedded Linux. 2001.

Marcos, C.J.G. 2006. GNU Linux. 2006.

Marzal, A. and Luengo, I.G. 2002. Introducción a la programación con Python y C. 2002.

Moore, J.S. 1992. A formal model of asynchronous communication and its use in mechanically. s.l. : National Aeronautics and Space Administration, Office of Management,, 1992.

Mueller, J.P. 2002. Aprendiendo Microsoft Windows XP en 21 lecciones avanzadas. 2002.

Oscar, S. 2013. Visual Paradigm for Uml. 2013.

Penin, A.R. 2012. Sistemas SCADA. s.l. : Marcombo, 2012.

Royo, J. 2004. Diseño digital. 2004.

Rumbaugh, J. and Booch, G. and Jacobson, I. 2007. El Lenguaje unificado de modelado: manual de referencia. 2007.

Shneiderman, B. and Plaisant, C. 2010. Designing the User Interface: Strategies for Effective Human-computer Interaction. 2010.

Sommerville, I. 2015. Software Engineering. 2015.

Un módulo de servicios públicos de procesos y sistemas multiplataforma. [Online] [Cited: 10 29, 2018.] https://code.google.com/p/psutil/.

ISBN: 978-959-16-4235-6